UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/725,016	12/02/2003	Serguei M. Beloussov	2230.0020000/MBR/GSB	3183

54089 7590 08/21/2006

BARDMESSER LAW GROUP, P.C.
910 17TH STREET, N.W.
SUITE 800
WASHINGTON, DC 20006

EXAMINER	
NGUYEN, PHILLIP H	

ART UNIT	PAPER NUMBER
2194	

DATE MAILED: 08/21/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/725,016	BELOUSSOV ET AL.	
	Examiner	Art Unit	
	Phillip H. Nguyen	2194	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 December 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-71* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-71* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *02 December 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the original filing of December 02, 2003. Claims 1-71 are pending and have been considered below.

Claim Objections

2. Claims 54, 59, 60, and 64 are objected to because of the following informalities:

Claim 54 should depend on claim 53.

Claim 59 should depend on claim 58.

Claim 60 should depend on claim 59.

Claim 64 should depend on claim 63.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claims 2, 3, and 26 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

5. Claims 2 and 3 refer to as in claim 1 and recite the limitation "prior to the allocating step". There is insufficient antecedent basis for this limitation in this claim. It is unclear to the Examiner whether the applicant interprets; "the allocating step" is part of the copying step as in claim 1 or "the allocating step" is the "copying step" as in claim 1. For the examining purposes, the Examiner interprets it as "the copying step".

Claim 26 recites the limitation "Intel X32 architecture". In the specification, the applicant dose not disclosed that the invention is going to use Intel X32 architecture. Therefore, it is unclear to the Examiner whether the applicant interprets; "Intel X32 architecture" is the same as "Intel X86 architecture". For the examining purposes, the Examiner interprets "Intel X32 architecture" as "Intel X86 architecture".

6. The applicant appears to invoke 35 U.S.C. 112 6th paragraph in claims 53, 61-63, 65, 67, 68 by using "means-plus-function" language. However, the Examiner notes that the claims recite sufficient structure, which is computer program code for performing those functions. While the claims pass the first of the three-prong test used to determine invocation of paragraph 6, since it also recites sufficient structure within the claims itself to perform entirely recited functions, the claim is not in means-plus-function format, even if the claim uses the term "means." Therefore, 35 U.S.C. 112 6th paragraph has not been invoked when considering these claim below.

7. The applicant also appears to invoke 35 U.S.C. 112 6th paragraph in claims 69-71 by using "means-plus-function" language. However, the Examiner notes that the only "means" for performing these cited functions in the specification appears to be computer program codes. While the claims pass the first of the three-prong test used to determine invocation of paragraph 6, since no other specific structural limitations are disclosed in the specification, the claims do not meet the other tests of the three prongs test. Therefore, 35 U.S.C. 112 6th paragraph has not been invoked when considering these claims below.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

9. Claims 1, 3, 6, 7, 9, 13-17, 19-32, 35-40, 42-45, 48-52, 54-58, 61-71 are rejected under 35 U.S.C. 102(b) as being anticipated by Mahalingaiah et al (US 5,983,337).

10. Claim 1: Mahalingaiah discloses a method of on-the-fly patching of executable code comprising:

a. Identifying original instructions to be changed (Col 5, line 45-50);

b. Copying the original instructions to a storage location (Col 5, line 21-45);

c. Adding a jump (branch) instruction to the copied instructions to return to a next instruction after the original instructions (Col 6, line 6-20); and

d. Replacing the original instructions with mark instructions and a transfer of control to a hook (Col 5, line 50-67; Col 6, line 1-5).

Claim 3: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above; and further discloses prior to the allocating step, masking interrupts (Col 23, line 20-40).

Claim 6: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above; and further discloses the original instructions are changed in reverse order (Col 26, line 7-28).

Claim 7: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above; and further discloses the mark instructions are the same length, in bytes, as the instructions to be changed (Col 6, line 19-45).

Claim 9: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above; and further discloses the modified instructions include a resolver to determine a number of the instructions at a location of the original code that had already been executed (Col 6, line 19-45).

Claim 13: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above; and further discloses enabling functionality of the copied instructions at the storage location (Col 5, line 46-60).

Claim 14: Mahalingaiah discloses a method as in claim 13 above; and further discloses the enabling step comprises reconciling addressing in the instructions in the storage location (Col 6, line 50-67).

Claim 15: Mahalingaiah discloses a method as in claim 14 above; and further discloses the enabling step comprises alignment of instructions in the instructions at the storage location (Col 6, line 5-20).

Claim 16: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above; and further discloses verifying that the original code is susceptible (valid) to patching (Col 16, line 42-57).

Claim 17: Mahalingaiah discloses a method as in claim 16 above; and further discloses verifying step determines whether any mark instructions are already present in the original instructions (Col 5, line 46-67).

Claim 19: Mahalingaiah discloses a method as in claim 16 above; and further discloses the verifying step determines whether the original instructions include a suitable jump point that can be modified to the transfer of control to the hook (Col 6, line 5-20).

Claim 20: Mahalingaiah discloses a method as in claim 16 above; and further discloses the verifying step determines whether the original instructions represent valid instructions (Col 6, line 20-45).

Claim 21: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above; and further discloses placing the hook in the memory (Col 5, line 47-55).

Claim 22: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above; and further discloses the hook has been previously placed in memory (Col 5, line 47-55).

Claim 23: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above; and further discloses the replacing step use an atomic write to replace the original instructions (Col 6, line 20-47).

Claim 24: Mahalingaiah discloses the method as in claim 23 above; and further discloses the atomic write replaces one instruction at a time (Col 17, line 15-30).

Claim 25: Mahalingaiah discloses the method as in claim 23 above; and further discloses the atomic write replaces multiple instructions at a time (Col 17, line 15-30).

Claim 26: mahalingaiah discloses the method as in claim 23 above; and further discloses for Intel X32 architecture, the atomic write uses any of "xchg," "lock cmpxchg8b," "lock cmpxchg," and "lock xchg" instructions (Col 28, line 25-67; Table 1).

Claim 27: Mahalingaiah discloses a method of on-the-fly patching of executable code comprising:

a. Verifying that original instructions to be changed are (Col 5, line 45-50);

b. Generating pseudooriginal code from the original instructions at different storage location from the original instructions (Col 5, line 21-45);

c. Adding a jump (branch) instruction to the pseudooriginal code to return to a next instruction after the original instructions (Col 6, line 6-20); and

d. Replacing the original instructions with tag instructions that indicate only their execution and a transfer of control to a hook (Col 5, line 50-67; Col 6, line 1-5).

Claim 28: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 27 above; and further discloses the original instructions are changed in reverse order (Col 26, line 7-28).

Claim 29: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 27 above; and further discloses the tag instructions are the same length, in bytes, as the instructions to be changed (Col 6, line 19-45).

Claim 31: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 27 above; and further discloses the modified instructions include a resolver to determine a number of the instructions at a location of the original code that had already been executed (Col 6, line 19-45).

Claim 32: Mahalingaiah discloses a method as in claim 31 above; and further discloses the resolver determines a number of instructions that had already been executed using mark instructions (Col 6, line 19-45).

Claim 35: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 27 above; and further discloses reconciling addressing in the instructions in the storage location (Col 6, line 50-67).

Claim 36: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 27 above; and further discloses verifying that the original code is susceptible to patching (Col 16, line 42-57).

Claim 37: Mahalingaiah discloses a method as in claim 36 above; and further discloses verifying step determines whether any tag instructions are already present in the original instructions (Col 5, line 46-67).

Claim 38: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 27 above; and further discloses placing the hook in the memory (Col 5, line 47-55).

Claim 39: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 27 above; and further discloses the replacing step use an atomic write to replace the original instructions (Col 6, line 20-47).

Claim 40: Mahalingaiah discloses a method of on-the-fly patching of executable code comprising:

a. Identifying original instructions to be changed (Col 5, line 45-50);

b. Allocating a storage location for storing a functionally equivalent copy of the original instructions (Col 5, line 20-45).

c. Copying the original instructions to a storage location (Col 5, line 21-45);

d. Replacing the original instructions with mark instructions and a transfer of control to a hook (Col 5, line 50-67; Col 6, line 1-5).

Claim 42: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 40 above; and further discloses adding a jump (branch) instruction to the copied instructions to return to a next instruction after the original instructions (Col 6, line 6-20).

Claim 43: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 40 above; and further discloses the original instructions are changed in reverse order (Col 26, line 7-28).

Claim 44: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above; and further discloses the modified instructions include a resolver to determine a number of the instructions at a location of the original code that had already been executed (Col 6, line 19-45).

Claim 45: Mahalingaiah discloses a method as in claim 44 above; and further discloses the resolver (scanning unit) determines a number of instructions that had already been executed using mark instructions (Col 6, line 19-45).

Claim 48: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 45 above; and further discloses verifying that the original code is susceptible to patching (Col 16, line 42-57).

Claim 49: Mahalingaiah discloses a method as in claim 48 above; and further discloses verifying step determines whether any mark instructions are already present in the original instructions (Col 5, line 46-67).

Claim 51: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 40 above; and further discloses the replacing step use an atomic write to replace the original instructions (Col 6, line 20-47).

Claim 52: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 40 above; and further discloses enabling functionality of the copied instructions at the storage location (Col 5, line 46-60).

Claim 53: Mahalingaiah discloses a computer program product for on-the-fly patching of executable code, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling at least one processor, the computer logic comprising:

a. Computer program code means for identifying original instructions to be changed (Col 5, line 45-50);

b. Computer program code means for copying the original instructions to a storage location (Col 5, line 21-45);

c. computer program code means for adding a jump (branch) instruction to the copied instructions to return to a next instruction after the original instructions (Col 6, line 6-20); and

d. computer program code means for replacing the original instructions with mark instructions and a transfer of control to a hook (Col 5, line 50-67; Col 6, line 1-5).

Claim 54: Mahalingaiah discloses a computer program product for on-the-fly patching of executable code, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling at least one processor as in claim 53 above; and further discloses the original instructions are changed in reverse order (Col 26, line 7-28).

Claim 55: Mahalingaiah discloses a computer program product for on-the-fly patching of executable code, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling at least one processor as in claim 53 above; and further discloses the mark instructions are the same length, in bytes, as the instructions to be changed (Col 6, line 19-45).

Claim 57: Mahalingaiah discloses a computer program product for on-the-fly patching of executable code, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling at least one processor as in claim 53 above; and further discloses the modified instructions include a resolver to determine a number of the instructions at a location of the original code that had already been executed (Col 6, line 19-45).

Claim 58: Mahalingaiah discloses a computer program product as in claim 57 above; and further discloses the resolver (scanning unit) determines a number of instructions that had already been executed using mark instructions (Col 6, line 19-45).

Claim 61: Mahalingaiah discloses a computer program product for on-the-fly patching of executable code, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling at least one processor as in claim 53 above; and further discloses computer program code

means for enabling functionality of the copied instructions at the storage location (Col 5, line 46-60).

Claim 62: Mahalingaiah discloses a computer program product as in claim 61 above; and further discloses the computer product code means for enabling functionality of the copied instructions at the storage location comprises computer product code means for reconciling addressing in the instructions in the storage location (Col 6, line 50-67).

Claim 63: Mahalingaiah discloses a computer program product for on-the-fly patching of executable code, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling at least one processor as in claim 53 above; and further discloses computer program code means for verifying that the original code is susceptible to patching (Col 16, line 42-57).

Claim 64: Mahalingaiah discloses a computer program product as in claim 63 above; and further discloses computer program code means for verifying determines whether any mark instructions are already present in the original instructions (Col 5, line 46-67).

Claim 65: Mahalingaiah discloses a computer program product for on-the-fly patching of executable code, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling at least one processor as in claim 53 above; and further discloses placing the hook in the memory (Col 5, line 47-55).

Claim 66: Mahalingaiah discloses a computer program product for on-the-fly patching of executable code, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling at least one processor as in claim 53 above; and further discloses the computer program code means for replacing step use an atomic write to replace the original instructions (Col 6, line 20-47).

Claim 67: Mahalingaiah discloses a computer program product for on-the-fly patching of executable code, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling at least one processor, the computer logic comprising:

a. Computer program code means for verifying that original instructions to be changed are susceptible to patching (Col 16, line 42-57);

b. Computer program code means for generating pseudooriginal code from the original instructions at a different storage location from the original instructions (Col 5, line 21-45);

c. Computer program code means for adding a jump (branch) instruction to the pseudooriginal code to return to a next instruction after the original instructions (Col 6, line 6-20); and

d. Computer program code means for replacing the original code with tag instructions that indicate only their execution and a transfer of control to a hook (Col 5, line 50-67; Col 6, line 1-5).

Claim 68: Mahalingaiah discloses a computer program product for on-the-fly patching of executable code, the computer program product comprising a computer useable medium having computer program logic recorded thereon for controlling at least one processor, the computer logic comprising:

a. Computer program code means for identifying original instructions to be changed (Col 5, line 45-50);

b. Computer program code means for allocating a storage location for storing a functionality copy of the original instructions (Col 5, line 20-45);

c. Computer program code means for copying the original instructions to the storage location (Col 5, line 21-45); and

d. computer program code means for replacing the original instructions with mark instructions and a transfer of control to a hook (Col 5, line 50-67; Col 6, line 1-5).

Claim 69: Mahalingaiah discloses a system for on-the-fly patching of executable code comprising:

a. Means for identifying original instructions to be changed (Col 5, line 45-50);

b. Means for copying the original instructions to a storage location (Col 5, line 21-45);

c. Means for adding a jump (branch) instruction to the copied instructions to return to a next instruction after the original instructions (Col 6, line 6-20); and

d. Means for replacing the original instructions with mark instructions and a transfer of control to a hook (Col 5, line 50-67; Col 6, line 1-5).

Claim 70: Mahalingaiah discloses a system for on-the-fly patching of executable code comprising:

a. Means for verifying that original instructions to be changed are susceptible to patching (Col 16, line 42-57);

b. Means for generating pseudooriginal code from the original instructions at a different storage location from the original instructions (Col 5, line 21-45);

c. Means for adding a jump (branch) instruction to the pseudooriginal code to return to a next instruction after the original instructions (Col 6, line 6-20); and

d. Means for replacing the original code with tag instructions that indicate only their execution and a transfer of control to a hook (Col 5, line 50-67; Col 6, line 1-5).

Claim 71: Mahalingaiah discloses a system for on-the-fly patching of executable code comprising:

a. Means for identifying original instructions to be changed (Col 5, line 45-50);

b. Means for allocating a storage location for storing a functionally equivalent copy of the original instructions (Col 5, line 20-45);

c. Means for copying the original instructions to a storage location (Col 5, line 21-45);

d. Means for replacing the original instructions with mark instructions and a transfer of control to a hook (Col 5, line 50-67; Col 6, line 1-5).

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. Claims 5, 8, 10, 11, 30, 33, 46, 56, and 59 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah et al (US 5,983,337).

Claim 5: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above, but does not explicitly disclose after the replacing step, unmasking interrupts. However, it would have been obvious to one having ordinary skill

in the art at the time of the invention was made to recognize that after the replacing step, unmasking interrupts to allow the owner to see and identifying the exception errors. Therefore, one would have been motivated to include unmasking interrupts step in order to recognize the exception errors.

Claims 8, 30, 56: Mahalingaiah discloses a method and computer program product of on-the-fly patching of executable code as in claim 1, 27, 53 above, respectively, but does not explicitly disclose wherein the mark instructions are shorter in length, in bytes, as the instructions to be changed, and include NOP (no operation) filler. The applicant discloses that in the Intel X86 architecture in the specification, when an instruction is changed, its length is never increased (Paragraph 49). It would have been obvious to one having ordinary skill in the art at the time the invention was made to recognize that in Intel X86 architecture, any changed command is either the same length as the original instruction or is shorter with corresponding NOP instructions (no operation) in the remaining bytes. Therefore, one would have been motivated to use this feature in the Intel X86 architecture for checking to see if the mark instructions are shorter in length, in bytes, as the instructions to be changed or any other useful reason for the invention.

Claim 10: Mahalingaiah discloses a method as in claim 9 above, but does not explicitly disclose the resolver determines a number of instructions that had already been executed using mark instructions. The applicant discloses that in Intel X86

architecture in the specification, the resolver always know how many instructions remain to be executed by using the mark instructions (Paragraph 60). It would been obvious to one having ordinary skill in the art at the time the invention was made to recognized that if the number of instructions remain to be executed are always known by using the mark instructions then the number of instruction that had already been executed by using mark instructions are also known. Therefore, one would have been motivated to use the mark instructions to determine the number of mark instructions had already been executed.

Claims 11, 33, 46, and 59: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 10, 32, 45, 58 above, respectively, but does not explicitly disclose if the number of instructions that had already been executed is less than a number of original instructions to be changed, the resolver calls the copied instructions at the storage location so as to imitate a "no patch installed" scenario. However, it would have been obvious to one having ordinary skill in the art at the time of the invention was made to recognize that after the instructions had already been executed, the contents of the instructions had already been changed. So, if there is an interrupt occurs during the execution process, the hook is going to execute before the unchanged instructions of the original code not the changed instructions. Otherwise the patching result would be different than expected. Therefore, one would have been motivated to add this step in the patching process to ensure the patching results are correctly.

Claim 12, 34, 47, and 60: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 11, 33, 46, and 59 above, respectively; and further discloses after execution of the instructions at the storage location, the resolver returns control to the next instruction (Col 18, line 56-62).

13. Claims 2, 4, 18, 41, and 50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah et al (US 5,983,337) in view of Scott et al (US 6,615,329).

Claim 2: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above, but does not explicitly disclose allowing a write operation on a page in memory where the original code is located. However, Scott discloses allowing a write operation on a page in memory (Col 3, line 30-37). It would have been obvious to one having ordinary skill in the art at the time the invention was made to consider allowing a write operation on a page in memory where the original code is located to copy data from the main memory to a storage location in order to perform the patching process. Therefore, in order to perform the patching process, one would have been motivated to allow a write operation on a page in memory to allow copying data from main memory to storage area.

Claim 4: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 1 above, but does not explicitly disclose disallowing (disabling) a write operation on the page in memory where the block of code is located. However, Scott discloses disable (disallow) a write operation on the page in memory where the block of code is located to protect the area from unauthorized user (Col 9, line 44-67). It would have been obvious to one having ordinary skill in the art at the time the invention was made to consider protecting the memory area by disable or disallow a write operation after data have been copied from memory to storage location.

Claims 18: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claims 16 above, but does not explicitly disclose determine whether any copy protect instructions are already present in the original instructions. However, Scott discloses instructions within the protected area of memory (Col 5, line 18-26). It would have been obvious to one having an ordinary skill in the art at the time the invention was made to recognize having instructions on the protected area to prevent the instructions from being modified by an unauthorized user. Therefore, one would have been motivated to have instructions located in the protected area of memory for protecting from modification.

Claim 41: Mahalingaiah discloses a method of on-the-fly patching of executable code as in 40 above, but does not explicitly disclose allowing a write operation on a page in memory where the original code is located. However, Scott discloses allowing

a write operation on a page in memory (Col 3, line 30-37). It would have been obvious to one having ordinary skill in the art at the time the invention was made to consider allowing a write operation on a page in memory where the original code is located to copy data from the main memory to a storage location in order to perform the patching process. Therefore, in order to perform the patching process, one would have been motivated to allow a write operation on a page in memory to allow copying data from main memory to storage area.

Claim 50: Mahalingaiah discloses a method of on-the-fly patching of executable code as in claim 48 above, but does not explicitly disclose determine whether any copy protect instructions are already present in the original instructions. However, Scott discloses instructions within the protected area of memory (Col 5, line 18-26). It would have been obvious to one having an ordinary skill in the art at the time the invention was made to recognize having instructions on the protected area to prevent the instructions from being modified by an unauthorized user. Therefore, one would have been motivated to have instructions located in the protected area of memory for protecting from modification.

Conclusion

14. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

a. Agarwal (US 6,305,010) discloses test, protection, and repair through binary code augmentation.

b. Tinker (US 6,948,164) discloses method and system for modifying executable code to add additional functionality.

c. Duesterwald et al (US 6,928,536) discloses dynamic execution layer interface for replacing instructions requiring unavailable hardware functionality with patch code and caching.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Phillip H. Nguyen whose telephone number is (571) 270-1070. The examiner can normally be reached on Monday - Friday 10:00 AM - 3:00 PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, James Myhre can be reached on (571) 270-1065. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

P.N

PN
8/10/06

James W. Myhre
Supervisory Patent Examiner